For this assignment you will be creating a barebones window using the Windows API and then setting up a rudimentary rendering system that will be implemented using D3D11. Follow the provided steps and ask for help or clarification on anything you need.

1. Walkthrough: Create a traditional Windows Desktop application

- Read and complete this tutorial on creating a simple window using the Windows API.
  - https://docs.microsoft.com/en-us/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp
- Test your code from the tutorial before proceeding

2. Modify Your Window Code

- Remove the case for handling the WM_PAINT message in the WndProc.
- Change the initial window dimensions to have a 16:9 aspect ratio.
- Modify WinMain to Launch a Console Window
  - Add this code Immediately before the call to CreateWindow.
  - This will spawn a console at start-up with our window when in debug mode.

```
#ifndef NDEBUG
AllocConsole();
FILE* new_std_in_out;
freopen_s(&new_std_in_out, "CONOUT$", "w", stdout);
freopen_s(&new_std_in_out, "CONIN$", "r", stdin);
std::cout << "Hello world!\n";          // Don't forget to include <iostream>
#endif
```

- Add the following code immediately before the final return in the WinMain function:

```
#ifndef NDEBUG
FreeConsole();
#endif
```

- Modify your main message loop to be nonblocking.
  - The default main message loop uses a blocking function call, GetMessage(...). This will cause the window to only update if an event/message is received. For complex games where battery life is not a concern, a non-blocking loop is preferable. This allows us to update as quickly as possible and to maximize our use of the hardware. Replace the main message loop as follows.

```
// Main message loop:
~~while (GetMessage(&msg, nullptr, 0, 0))~~
~~{~~
    ~~TranslateMessage(&msg);~~
    ~~DispatchMessage(&msg);~~
~~}~~
```

```
// Main application loop:
while (true)
{
    // Process all messages, stop on WM_QUIT
    if(PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
    {
            // WM_QUIT does not need to be // translated or dispatched
            if (msg.message == WM_QUIT)
                    break;
            // Translates messages and sends them to WndProc
            TranslateMessage(&msg);
            DispatchMessage(&msg);
    }
    else
    {
    // In the future, do per frame/tick updates here...
    }
}
```

3. Review the Provided Base Code

- math_types.h
  - Provides flexible definitions for vector and matrix types.
- view.h
  - Defines a simple “view” type. Think of this as a generalization of a camera class. Can be used for a camera, or another type of view like a shadowmapping light source. This type will be very simple for now.
- renderer.h
  - Defines the interface for the renderer. This interface is simple, high-level, and platform independent. It should have no knowledge of graphics API specific functions and types. This type will be very simple for now.
- d3d11_renderer_impl.h
  - Defines the implementation for the renderer. The majority of your remaining work for this assignment will be done here. This header must only be included by renderer.cpp.

(CONTINUED)

Implement the Renderer

- Create an instance of the renderer class within Winmain, before entering the main loop.
- Within the main loop, if no message is processed then have the renderer draw the scene.
- Setup the D3D11 Renderer Implementation
- Within d3d11_renderer_impl.h complete the following steps
  - Create Device and Immediate Context
    - Clear the backbuffer to a color other than white or black and present it
    - Reference:
    - How To: Create a Device and Immediate Context
    - https://msdn.microsoft.com/en-us/library/windows/desktop/ff476879(v=vs.85).aspx

  - Create Vertex and Pixel Shaders
    - Use vs_cube.hlsl and ps_cube.hlsl
    - Reference:
    - Compiling Shaders
    - https://msdn.microsoft.com/en-us/library/windows/desktop/bb509633(v=vs.85).aspx
    - ID3D11Device::CreateVertexShader method
    - https://msdn.microsoft.com/en-us/library/windows/desktop/ff476524(v=vs.85).aspx

  - Create Constant Buffer
    - Reference:
    - How to: Create a Constant Buffer
    - https://msdn.microsoft.com/en-us/library/windows/desktop/ff476896(v=vs.85).aspx